# 常用操作系统命令

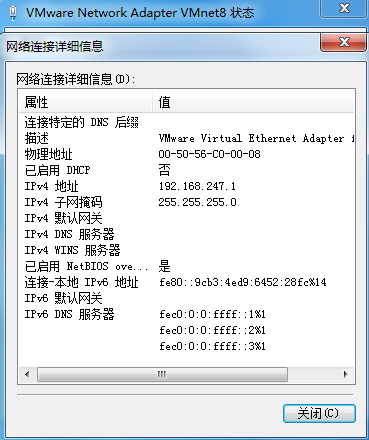
## Ubuntu

### 图像和字符启动界面设置

查看版本信息

### vmware下Ubuntu16 nat方式上网

#### 主机Net8 信息(NAT网络模式)



设置网关为

192.168.247.2

#### 客户机网络配置

ubuntu的网络配置信息放在 /etc/network/interfaces 中

1.如果配置动态获取ip，则在上述文件中加入以下内容：

     auto ens33

     iface ens33 inet dhcp

zzh@zzh-virtual-machine: $ pwd

/proc/sys/net/ipv4/conf

zzh@zzh-virtual-machine:/proc/sys/net/ipv4/conf$ ls

all default docker0 ens33 lo veth96a833a

2.如果配置静态ip，则添加如下内容：

# interfaces(5) file used by ifup(8) and ifdown(8)

auto

iface ens33 inet static

address 192.168.247.8

netmask 255.255.255.0

gateway 192.168.247.2

3.配置dns服务器

ubuntu 的dns服务器信息，放在 /etc/resolv.conf中,添加dns服务器地址

nameserver 192.168.247.2 (这就是NAT配置信息里面的网关IP)

4.重启网络服务

sudo /etc/init.d/networking restart

### 软件包管理

#### 软件包搜索

* **apt-cache search foo** - 搜索和”foo”匹配的包。
* **apt-get remove 软件包名称** - 删除已安装的软件包（保留配置文件）
* **dpkg -L foo** - 显示名为“foo”的包都安装了哪些文件以及它们的路径，很有用的命
* **apt-cache show package\_name**

显示指定软件包的信息，包括版本号，安装状态和包依赖关系等.

* **apt-file search foo** - 类似dlocate和dpkg -S，但搜索所有有效软件包包，不单单只是你系统上的已安装的软件包。– 它所回答的问题是“哪些软件包提供这些文件”。你必须安装有apt-file软件包，并且确保apt-file数据库是最新的。

# Mysql

## Ubuntu16 安装Mysql

在5.7.18版本中，使用tar.gz安装时，也就是压缩包解压出来安装这种，已经不再需要my.cnf文件也能正常运行

安装依赖：

*apt-cache search libaio* # search for info

*apt-get install libaio1* # install library

<https://dev.mysql.com/doc/refman/5.7/en/binary-installation.html>

sudo groupadd mysql

sudo useradd -g mysql -s /bin/bash mysql -md /home/mysql

cd /usr/local

tar zxvf /path/to/mysql-VERSION-OS.tar.gz

ln -s full-path-to-mysql-VERSION-OS mysql

cd mysql

mkdir mysql-files

chown mysql:mysql mysql-files

chmod 750 mysql-files

bin/mysqld --initialize --user=mysql

bin/mysql\_ssl\_rsa\_setup

bin/mysqld\_safe --user=mysql &

SET PASSWORD = PASSWORD(‘ABC\_abc1’);

cp support-files/mysql.server /etc/init.d/mysql

export PATH=$PATH:/usr/local/mysql/bin

mysql安全启动(mysqld\_safe)

mysql\_safe 从配置文件中读取[mysqld],[server],[mysqld\_safe]等选项

mysqld\_save --user=mysql &

停止

mysqladmin -uroot -pxxx shutdown -S /tmp/mysq.sock

mysql(mysql.server)服务方式启动

mysql.server为一个shell脚本，他会调用mysqld\_safe，该脚本会读取配置文件的[mysql.server]以及[mysqld]选项。

 /etc/init.d/mysql start

Mysql配置解析



## SQL

### DDL

系统信息

select schema()/database()

select sysdate()/now();

select curdate();

select curtime();

select user();

select version();

show status;

show variables;

数据库

show databases \G

create database dbname

drop database dbname

use dbname;

表操作

create table tbname()

drop table tbname;

show tables;

desc tbname;

show create table tbname;

导入导出

mysqldump –u 用户名 -p 数据库名 > 导出的文件名

mysql

-p 后面有空格，表示数据库，无空格表示密码

字段操作

alter table tbname add age int(30);

alter table tbname modify id int(30);

alter table 表名 change 原字段名 新字段名 新字段类型;

alter table tbname drop age;

### DML

表复制

Like

As

记录操作

select \* from tbname where

insert into tbname values();

### DCL

存储过程

参数类型：

In

Inout

Out

变量：

1. 会话变量
2. 普通变量,无@符号

声明： declare

赋值: set （等号两边有空格） /select

Show procedure status \G

show create procedure test;

drop procedure test;

delimiter //

create procedure test(in p\_in int )

begin

select p\_in;

set p\_in=2;

select p\_in;

end //

delimiter ;

set @P\_IN=1;

call test(@P\_IN);

触发器

语法：create trigger 触发器名称 触发的时机 触发的动作

on 表名 for each row 触发器状态。

触发器名称：phptrigger 可以自己随意起

触发的时机： before /after 在执行动作之前还是之后

触发的动作

指的激发触发程序的语句类型

insert

update

delete

show triggers;

create trigger delCategory after delete on category for each row

-> delete from books where bTypeid='3';

show create trigger delCategory;

drop trigger delCategory;

事务

ACID

Start transaction

Commit

Set autocommit=0

Mysql架构体系

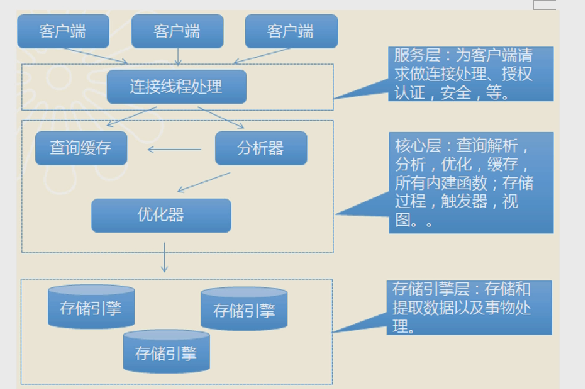
* 1. 主配置文件 /etc/my.cnf
  2. 进程通讯sock文件 socket= /var/lib/mysql/mysql.sock
  3. 日志文件 log-error=/var/log/mysqld.log
  4. 进程ID文件 pid-file=/var/run/mysqld/mysqld.pid
  5. 二进制日志文件 log-bin=
  6. 慢查询日志文件 log-slow-queries=
  7. 数据文件
  8. [root@xuegod63 ~]# ls /var/lib/mysql/book/
  9. books.frm books.MYI category.MYD db.opt t.MYD
  10. books.MYD category.frm category.MYI t.frm t.MYI
  11.  frm meta data 存储表定义
  12.  MYD 存储数据文件
  13.  MYI 存储索引文件

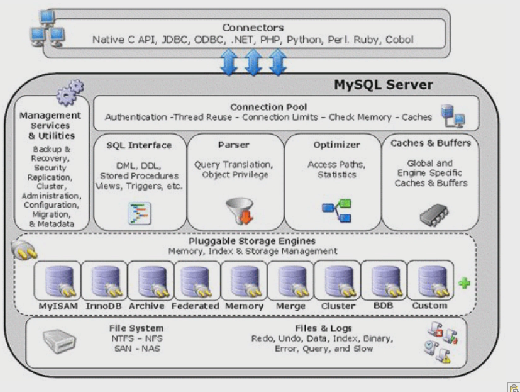
Mysql系统逻辑架构：

服务层：链接，授权，安全认证

核心层:解析，分析，优化，缓存

存储引擎层：存储和提取数据，事务处理





# Maven

## Maven镜像更换为阿里云中央仓库(精)

<mirror>

<id>nexus-aliyun</id>

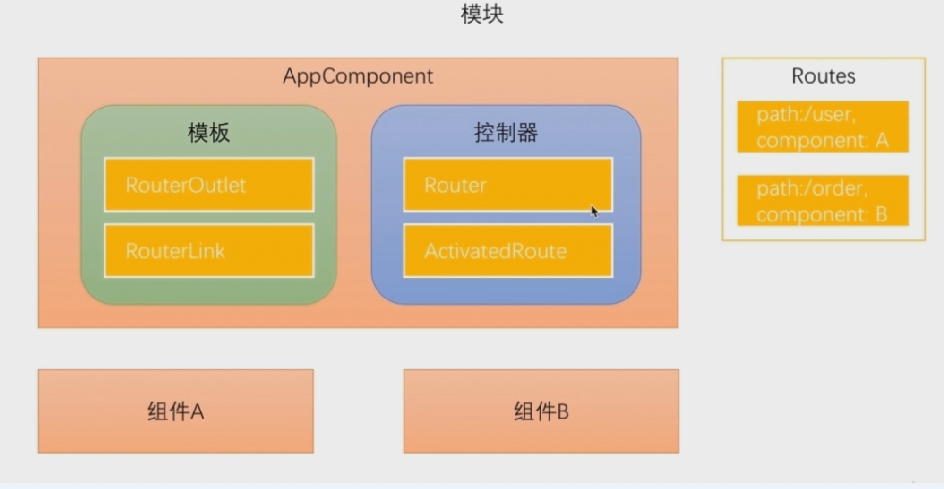
<mirrorOf>central</mirrorOf>

<name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>

# Angular6



启动过程

angular.json 项目配置文件

启动脚本：

main: main.ts

首页：

index: index.html

styles:

scripts:

## 集成Jquery

cnpm install jquery@3.1.1 –save

类型描述文件:（在typescript直接引用jquery）

cnpm install --save @types/jquery

修改angular.json

"styles": [

"node\_modules/jquery/dist/jquery.js",

"src/styles.css"

],

## 集成 Bootstrap

安装依赖包

cnpm install bootstrap@3.3.7 –save

cnpm install bootstrap --save

cnpm install --save @types/bootstrap

导入依赖包

修改angular.json

"node\_modules/bootstrap/dist/css/bootstrap.css"

"node\_modules/jquery/dist/jquery.js",

"node\_modules/bootstrap/dist/js/bootstrap.js"

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

"src/styles.css"

],

"scripts": [

"node\_modules/bootstrap/dist/js/bootstrap.min.js"

]

},

## 集成Jsplumb

## 集成echarts

路由

## 路由

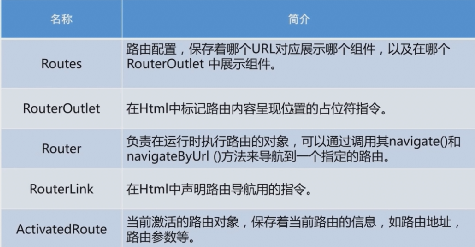
### 安装路由

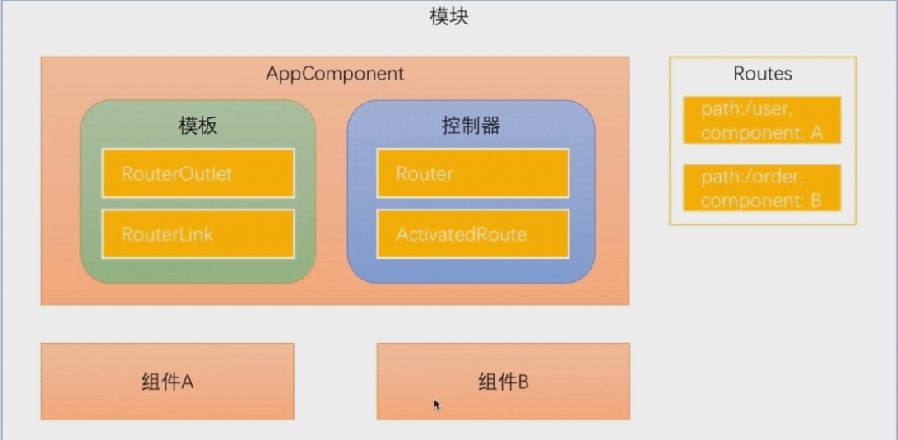
npm i --save @angular/router

以上命令执行后，将会自动下载 @angular/router 模块node\_modules 文件夹中。

ng g testset –routing

### 路由对象





### Routes：路由配置

RouterOutlet：路由内容呈现位置的占位符指令

导航：(路由转换)

控制器中使用

Router:

HTML中使用

RouterLink:

Href

href

ActivitedRoute: 当前激活的对象

Routes 存在模块中：

### 路由参数传递

路由时传递数据：

1查询参数

/product?id=1&name=zzh => ActivitedRoute.queryParam[id]

2路径参数

{/product/:id}

ActivitedRoute.params[id]

3路由配置中传递数据

### 组件间通讯

通讯机制：

input

output EvemtEmitter

路由

@ViewChild

service

中间人

* + Base href
* 使用路由
  + RouterModule.forRoot
  + RouterModule.forChild
* 配置路由
* Displaying routes
* Futher configuration
  + Dynamic routes
  + Child routes
  + Component-less routes
  + loadChildren
* 路由指令
  + routerLink
  + routerLinkActive
* 路由API

ng g c add-sample  --module=app

原因：

项目中有多个模块（包含appModule , routingModule等），

动态路由：

创建路由模块app.routing.module.ts

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { HomeComponent } from './home/home.component';

import { ProductsComponent } from './products/products.component';

const routes: Routes = [

/\*\*path不能以'/'开头 \*/

{

path: '', component: HomeComponent

},

{

path: 'product', component: ProductsComponent

}

]

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule],

providers: []

})

export class AppRoutingModule {

}

主模块中声明导入路由模块

@NgModule({

declarations: [

AppComponent,

ProductsComponent,

LoginComponent,

HomeComponent

],

imports: [

BrowserModule,

AppRoutingModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

配置路由访问(routerLink方式)和占位符

<!--The content below is only a placeholder and can be replaced.-->

<div style="text-align:center">

<h1>

Welcome to {{ title }}!

</h1>

<a [routerLink]="['/']">home page</a>

<a [routerLink]="['/product']">product detial</a>

<!-- <app-products></app-products> -->

<router-outlet></router-outlet>

</div>

配置路由访问(Router方式)

<!--The content below is only a placeholder and can be replaced.-->

<div style="text-align:center">

<h1>

Welcome to {{ title }}!

</h1>

<a [routerLink]="['/']">home page</a>

<a [routerLink]="['/product']">product detial</a>

<input type="button" value="product detail" (click)="toProductDetails()">

<!-- <app-products></app-products> -->

<router-outlet></router-outlet>

</div>

Component中主任router模块，通过navigate或者navigateByUrl实现跳转

import { Component } from '@angular/core';

import { Router } from '@angular/router';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'PCM';

constructor(private router: Router){

}

toProductDetails(){

this.router.navigate(['/product']);

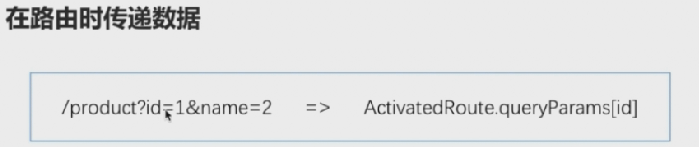
}

}

在查询参数中传递数据

在路由路径中传递数据

在路由配置中传递数据







ngOnInit() {

//参数快照

//this.productId = this.routeInfo.snapshot.queryParams["id"];

// this.productId = this.routeInfo.snapshot.params["id"];

//参数订阅

this.routeInfo.params.subscribe((params: Params) =>{this.productId=params["id"]});

console.log(this.productId);

}

参数快照不能解决路由到自身路由的问题，

因为ngOinit()只会加载一次，此时可以使用参数订阅解决，subscribe.

### 路由重定向

### 子路由

### 辅助路由

路由守卫：

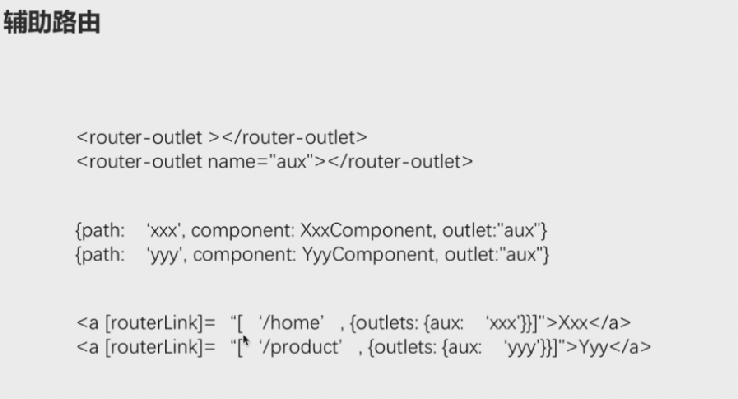
1、只有当用户已经登录并且拥有某些权限时才能进入某些路由

2、一个由多个表单组成的向导，例如注册流程，用户只有在当前路由中的组件中填写了满足要求的信息才可以导航到下一个路由

3、当用户未执行保存操作而试图离开当前导航时提醒用户

CanActivate：处理导航到某路由情况

CanDeactivate: 处理从当前路由离开情况



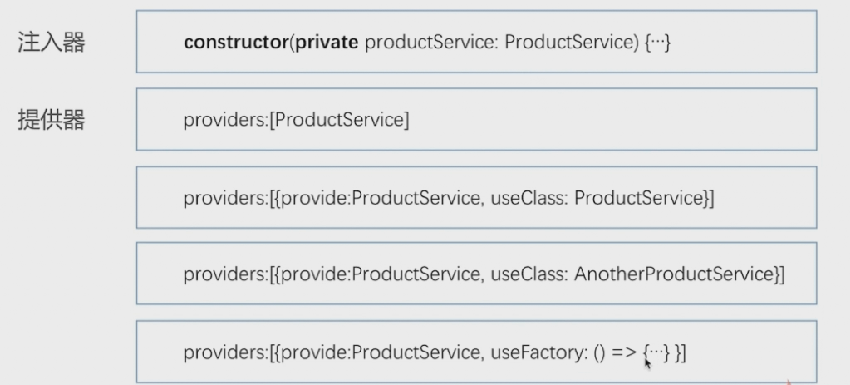
### Resolve守卫

进入组件之前，把组件需要的数据预先加载好

# 依赖注入

使用提供器



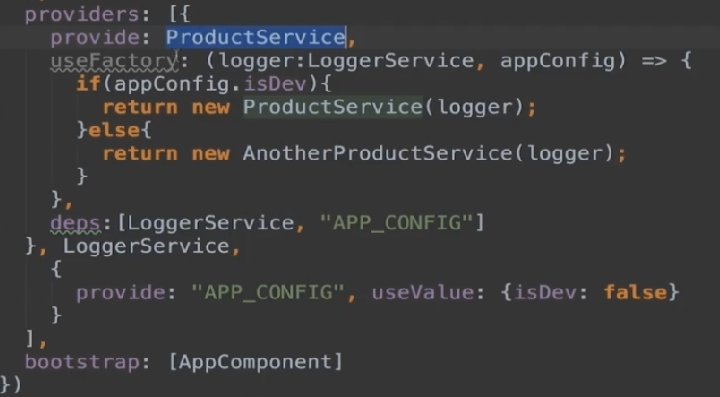


使用值

使用工厂方法

useClass

使用工厂和值声明提供器



默认单向绑定：

数据绑定：

{{}}

属性绑定

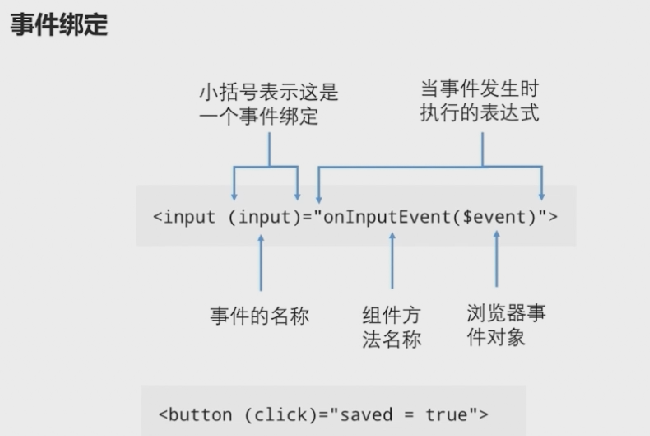
[attributeName]

Class绑定

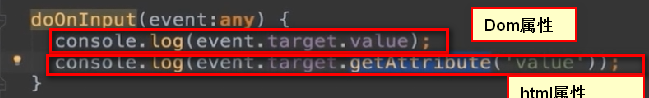
双向绑定

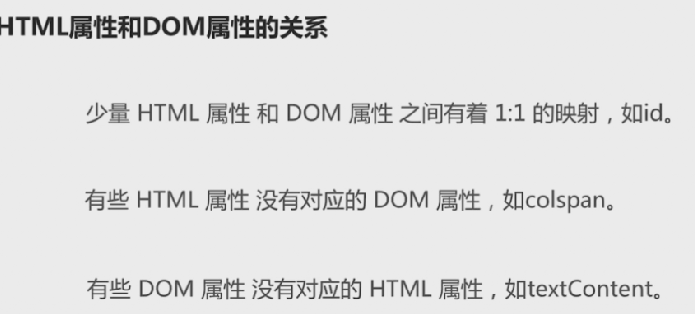
[(ngModel)]

事件绑定





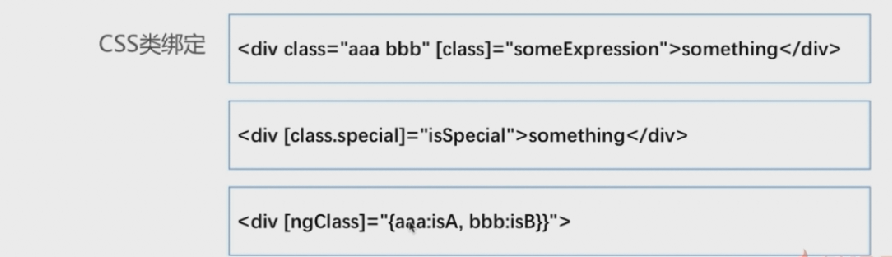


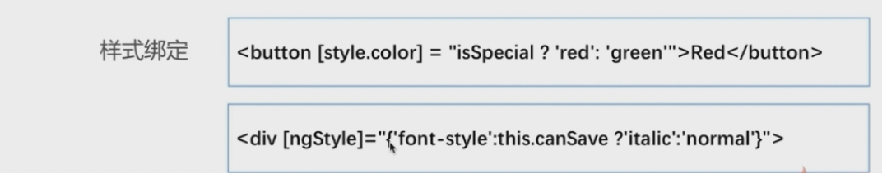




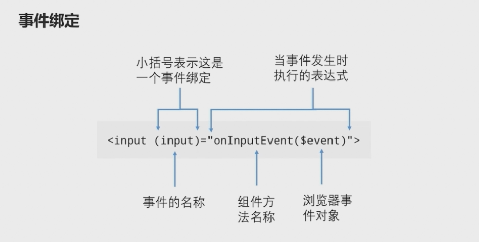
HTML属性绑定



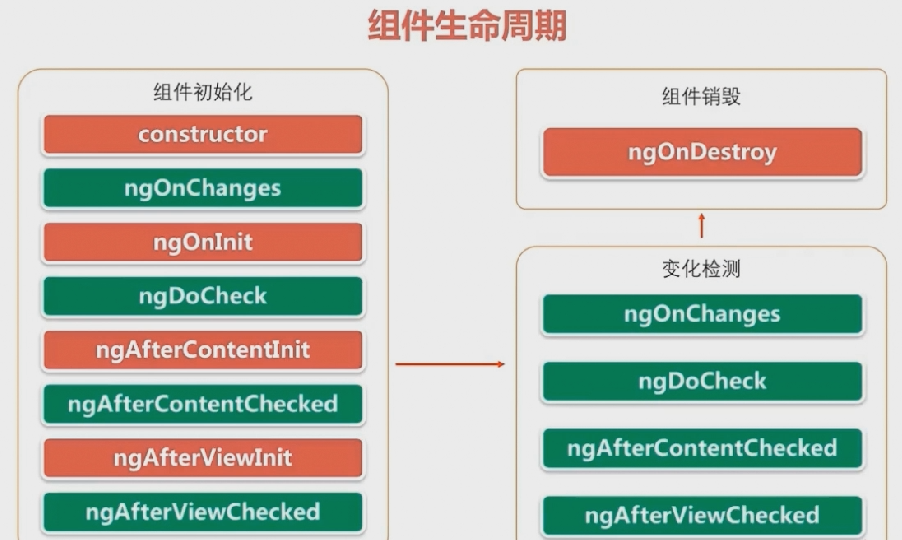




模板变量

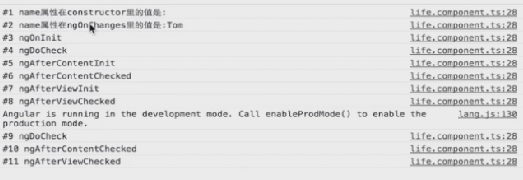


# 生命周期



红色的只会被调用一次

绿色的会被多次调用



## ogOnChange:

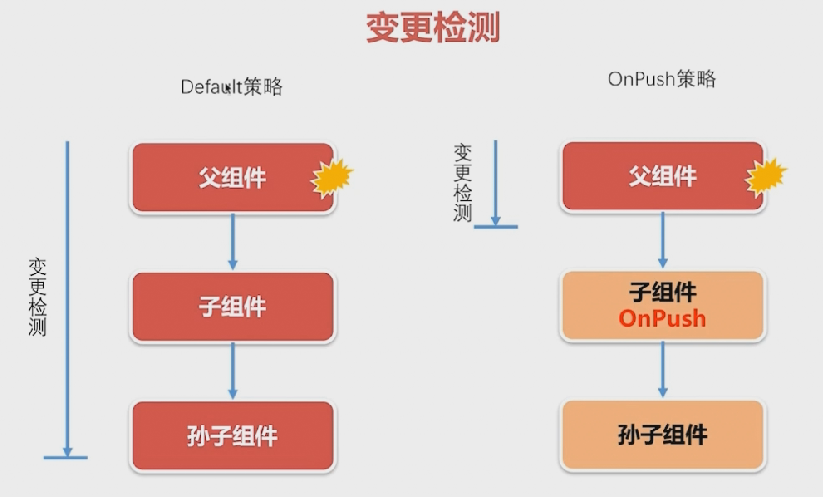
父组件修改或者初始化子组件时调用，如果组件没有input属性，那么不会被调用，只会检测引用地址，不会检测对象属性内容

## 变更检测机制：

docheck钩子，检测对象内容

变更检测时机：触发事件

依赖：zone.js 保证页面和属性时同步的。



## View钩子

<app-child #child1></app-child>

Component中调用子组件方法

@viewChild

Child1: ChildComponent;

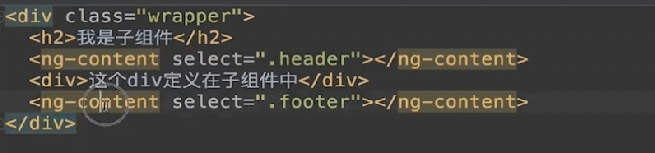
This.child1.functionName();

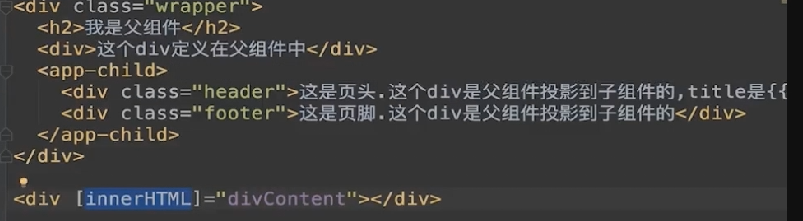
模板中调用子组件方法

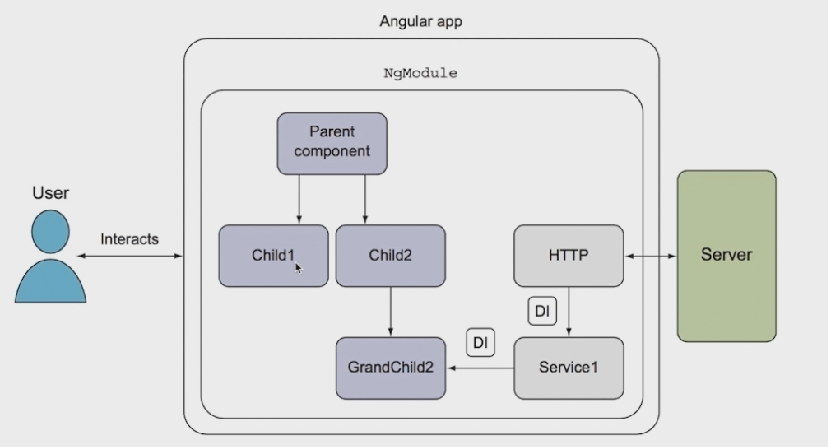
<button (click)=”child1.funcnamt()”>

Ng-content指令

innerHTML









数据绑定

1、页面到—》模型

事件绑定

2、模型到页面

{{}}

属性绑定

[属性]

样式绑定

[class.样式]

1. 双向绑定